```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

/*
    Problem # 2 Using LU Decomposition-Crout's Method
*/
double dblAbs(double num){
    num = sqrt(num*num);
    return num;
}

int main()
{
    double U[13][13], L[13][13];
    double b[13] = {2, 0, 0, 0, 4, 0, 0, 0, 2, 0, 0, 0, 0};
    double x[13], y[13], bsol[13];
    double a0[13][13] = {
    { 1.0250, -0.4000,  0.0000,    0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    {-0.4000,  1.9135, -0.2998,  0.0000,  0.0000, -1.0338, -0.1799,  0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000, -0.2998,  2.1415, -0.4417,  0.0000,  0.0000, -1.4000,  0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000,  0.0000, -0.4417,    1.0077,  0.0000,  0.0000,    0.0000, -0.5660,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    {-0.6250,  0.0000,  0.0000,    0.0000,  2.0500, -0.8000,    0.0000,   0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000, -1.0338,    0.0000,   0.0000, -0.8000,  3.8097, -0.4401,    0.0000,  0.0000, -1.0000, -0.0595,    -0.4764,  0.0000},
    { 0.0000, -0.1799, -1.4000,    0.0000,  0.0000, -0.4401,    4.0492, -0.7067,  0.0000,  0.0000,    0.0000, -1.1159, -0.2067},
    { 0.0000,  0.0000,    0.0000, -0.5660,  0.0000,  0.0000, -0.7067,    2.2161,  0.0000,  0.0000,    0.0000,   0.0000, -0.9433},
    { 0.0000,  0.0000,    0.0000,   0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  1.0250, -0.4000,    0.0000,   0.0000,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -1.0000,    0.0000,   0.0000, -0.4000,  2.0667, -0.6667,    0.0000,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -0.0595,    0.0000,   0.0000,  0.0000, -0.6667,    1.3788, -0.6526,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -0.4764, -1.1159,    0.0000,  0.0000,  0.0000, -0.6526,    2.5098, -0.2650},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000,  0.0000, -0.2067,   -0.9433,  0.0000,  0.0000,    0.0000, -0.2650,  1.4150},
    };
    double a[13][13] = {
    { 1.0250, -0.4000,  0.0000,    0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    {-0.4000,  1.9135, -0.2998,  0.0000,  0.0000, -1.0338, -0.1799,  0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000, -0.2998,  2.1415, -0.4417,  0.0000,  0.0000, -1.4000,  0.0000,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000,  0.0000, -0.4417,    1.0077,  0.0000,  0.0000,    0.0000, -0.5660,  0.0000,  0.0000,    0.0000,   0.0000,  0.0000},
    {-0.6250,  0.0000,  0.0000,    0.0000,  2.0500, -0.8000,    0.0000,   0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  0.0000},
    { 0.0000, -1.0338,    0.0000,   0.0000, -0.8000,  3.8097, -0.4401,    0.0000,  0.0000, -1.0000, -0.0595,    -0.4764,  0.0000},
    { 0.0000, -0.1799, -1.4000,    0.0000,  0.0000, -0.4401,    4.0492, -0.7067,  0.0000,  0.0000,    0.0000, -1.1159, -0.2067},
    { 0.0000,  0.0000,    0.0000, -0.5660,  0.0000,  0.0000, -0.7067,    2.2161,  0.0000,  0.0000,    0.0000,   0.0000, -0.9433},
    { 0.0000,  0.0000,    0.0000,   0.0000, -0.6250,  0.0000,    0.0000,   0.0000,  1.0250, -0.4000,    0.0000,   0.0000,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -1.0000,    0.0000,   0.0000, -0.4000,  2.0667, -0.6667,    0.0000,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -0.0595,    0.0000,   0.0000,  0.0000, -0.6667,    1.3788, -0.6526,  0.0000},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000, -0.4764, -1.1159,    0.0000,  0.0000,  0.0000, -0.6526,    2.5098, -0.2650},
    { 0.0000,  0.0000,    0.0000,   0.0000,  0.0000,  0.0000, -0.2067,   -0.9433,  0.0000,  0.0000,    0.0000, -0.2650,  1.4150},
    };


    double u0 = 5;
    int total, col, row, k, i, j,p;
    double dummy, m;

    total = sizeof(a)/sizeof(a[0][0]);      //
    col = sizeof(a[0])/sizeof(a[0][0]);     // Determines the number of rows and columns
    row = total/col;                        //

    for(i=0; i<row; i++){
```

```
        b[i] = b[i]*u0;
    }

    //Initialize the L and U Matrix to zero
    for (j=0; j<col; j++){
        for (i=0; i <row; i++){
            U[i][j] = 0;
            L[i][j] = 0;
        }
    }

    for (j=0; j<col; j++){
        U[j][j] = 1;
        for (i=0; i <row; i++){
            // Compute for elements of L
            if (i>=j){
                L[i][j] = a[i][j];
                for(k = 0; k < j; k ++){
                    L[i][j] = L[i][j] - L[i][k]*U[k][j];
                }
            }
        }
        for (i=0; i <row; i++){
            // Compute for elements of U
            if (i>j) {
                U[j][i] = a[j][i];
                for(k = 0; k<j; k++){
                    U[j][i] = U[j][i] - L[j][k]*U[k][i];
                }
                U[j][i] = U[j][i]/L[j][j];
            }
        }
    }

    for(i=0;i<row;i++){
        //Forward Substitution
        y[i] = b[i];
        for(k = 0; k < i; k ++) {
            y[i] = y[i] - L[i][k]*y[k];
        }
        y[i] = y[i]/L[i][i];
    }
    for(i=(row-1); i >= 0;i--){
        //Backward Substitution
        x[i] = y[i];
        for(k=(row-1); k > i; k --) {
            x[i] = x[i] - U[i][k]*x[k];
        }
    }

    printf("\nPotential at Each node are:\n");
    for(i=0;i<row;i++){
        printf("x[%d] = %0.9lf \n",i+1,x[i]);
    }

    //Checking:
    printf("\nB's are:\n");
    for(i=0;i<row;i++){
        bsol[i] = 0;
        for(j=0;j<row;j++){
            bsol[i] = bsol[i] + a0[i][j]*x[j];

        }
        printf("b[%d] : %0.9lf \n",i+1,bsol[i]);


    }

    return 0;
}
```